OpenBiblio System Administration

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Thanks for being an OpenBiblio system administrator. We're glad that you are helping your library in this way.

1 Configuring the staff interface

This section will describe settings found in the admin menu.

2 Configuring the OPAC

This section will describe settings found in the admin menu.

3 Managing plugins

OpenBiblio has a very flexible plugin system.

3.1 Creating a new plugin

This requires basic PHP skills and access to the files.

3.1.1 Design the plugin

1. Create a new directory that begins with the word plugin, followed by an underscore, followed by the name of your plugin. In UNIX systems, you might type something like this:

   ```
   mkdir plugin_excitingPlugin
   ```

2. If you need to include custom CSS or JS, add it to a file called custom_head.php in this new directory.
3. Create the PHP files you need. These can be called anything you want, but we recommend that your naming conventions are pretty similar to the rest of the OpenBiblio code.

Create a plugin server

Your plugin's server — located at `plugin_excitingPlugin/excitingSrvr.php` — is responsible for three tasks:

1. Performing any actions the user requests.
2. Obtaining any relevant data from the database, file structure, or external API.
3. Displaying the relevant data to the user in a nice HTML format.

A basic plugin server consists of a require statement to get the necessary data and methods and a switch statement to take care of the `$_REQUEST['mode']` variable. Here's a simple example:

```
<?php
        require_once('../shared/common.php');
        switch ($_REQUEST['mode']){
                case 'exciting':
                        echo 'This plugin is very exciting.';
                        break;
                case 'boring':
                        echo 'This plugin is pretty boring.';
                        break;
```

```
                default:
                        echo T('invalid mode');
                        break;
        }
```

If you would like to access data from the database, you should require the appropriate model from the model directory. For example, if you'd like to use the Biblios model, you would include this in your plugin_excitingPlugin/excitingSrvr.php:

```
require_once('../model/Biblios.php');
$bib = new Biblios;
```

You can then access all those great methods and data through the `$bib` object.

3.1.2 Add your plugin to the OpenBiblio menu

1. Create a new directory inside the plugins directory with the name of your plugin. In UNIX systems, you might type something like this:

   ```
   mkdir plugins/excitingPlugin
   ```

2. Create a file within this new directory called nav.nav

   ```
   <?php
   Nav::node('tools/excitingPlugin', 'Exciting Plugin', '../plugin_excitingPlugin/ ↩
       excitingPluginForms.php');
   ```

3. If you'd like to use Obib's localization framework but need to localize some more terms, create a file called tran.tran. You can use this to add new terms to your locale.

   Tip
   Plugins already have all of the exciting localization ability built in. However, note that you cannot override localized term definitions within tran.tran.

Warning
There is not yet a place to specify localized terms for plugins.

+ . Turn on your plugin using Tools → Plugin Manager.

4 Managing the MySQL database

This section will describe the database

5 Managing the server

This section will discuss other server-level considerations.

5.1 Logs

This section will mention logs of particular interest to OpenBiblio administrators.